Remaining Classification Techniques

1. **Support Vector Machines (SVM)**:
    - Linear Kernel
    - Polynomial Kernel (Quadratic, Cubic)
    - Radial Basis Function (RBF) Kernel
    - Sigmoid Kernel
2. **K-Nearest Neighbors (KNN)**:
    - Weighted KNN
    - Distance-Based KNN
3. **Discriminant Analysis**:
    - Linear Discriminant Analysis (LDA)
    - Quadratic Discriminant Analysis (QDA)
4. **Ensemble Methods**:
    - Bagging
    - AdaBoost
    - Gradient Boosting
    - XGBoost
    - LightGBM
    - CatBoost
    - Stacking Classifier
    - Voting Classifier (Hard and Soft Voting)

ADABOOST:

Pre-Processing

1. **Handling Missing or Irrelevant Columns:**
    - The column `Name` is dropped as it is irrelevant to the model.
2. **One-Hot Encoding:**
    - Categorical columns are identified and one-hot encoded using `pd.get_dummies`. This transforms categorical data into a numerical format required by machine learning algorithms.
3. **Feature and Target Separation:**
    - Features (`X`) and the target (`y`) are separated. The target is `Chronic Medical Conditions_Yes`.
4. **Train-Test Split:**
    - The dataset is split into training (70%), validation (15%), and test (15%) sets to evaluate the model effectively.
5. **Balancing the Dataset:**
    - Applied **SMOTE** (Synthetic Minority Oversampling Technique) to address class imbalance by generating synthetic samples for the minority class.

6. **Feature Scaling:**
    - Used `StandardScaler` to normalize features, ensuring all features contribute equally to the model's performance and optimization.

AdaBoost Technique

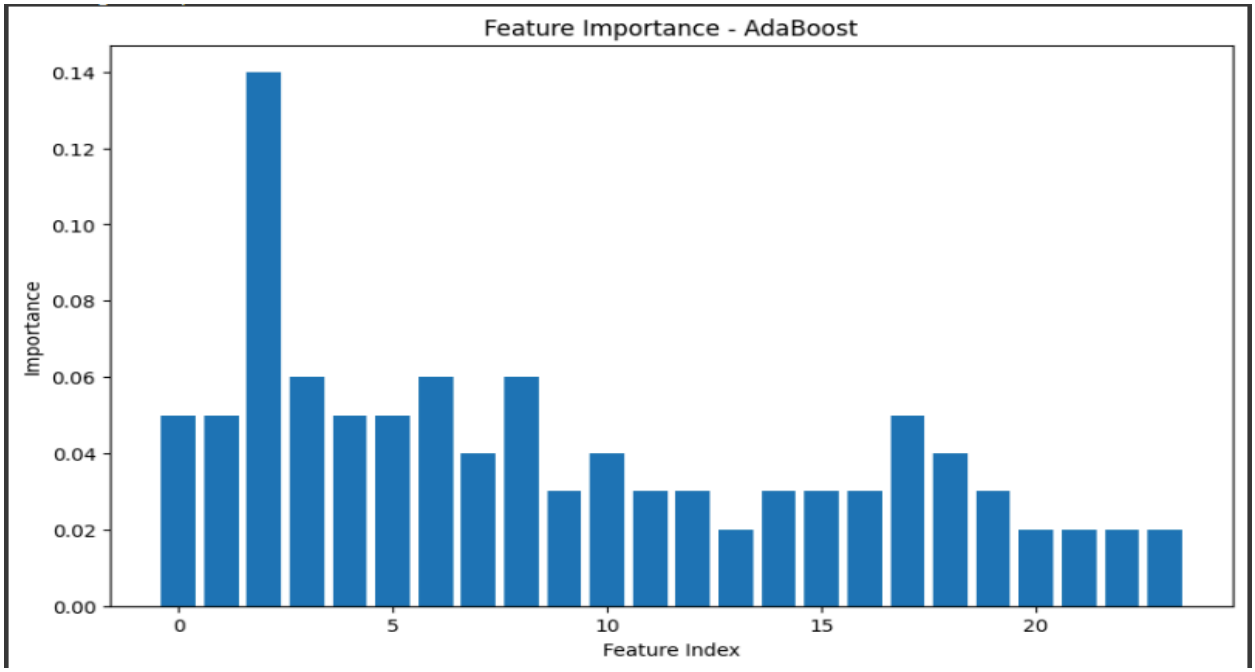
1. **Base Estimator:**
    - The base estimator for AdaBoost is a **Decision Tree** with a maximum depth of 1, making it a **stump**. This helps in focusing on learning weak classifiers to improve overall performance.
2. **Number of Estimators:**
    - The AdaBoost classifier is initialized with `n_estimators=100`. This defines the number of boosting rounds or weak learners to be combined.
3. **Random State:**
    - A fixed `random_state` ensures reproducibility.
4. **Training:**
    - The balanced and scaled training data is used to train the AdaBoost model.
5. **Feature Importance:**
    - The importance of each feature is computed and visualized, helping in identifying the key contributors to the model's performance.
6. **Training and Validation Error Curves:**
    - Errors on the training and validation datasets are plotted as the number of estimators increases. This helps in understanding the behavior of the model and potential overfitting or underfitting trends.

Focus on Important Features:

- These features likely capture critical information related to the target variable (e.g., Chronic Medical Conditions).

Dimensionality Reduction:

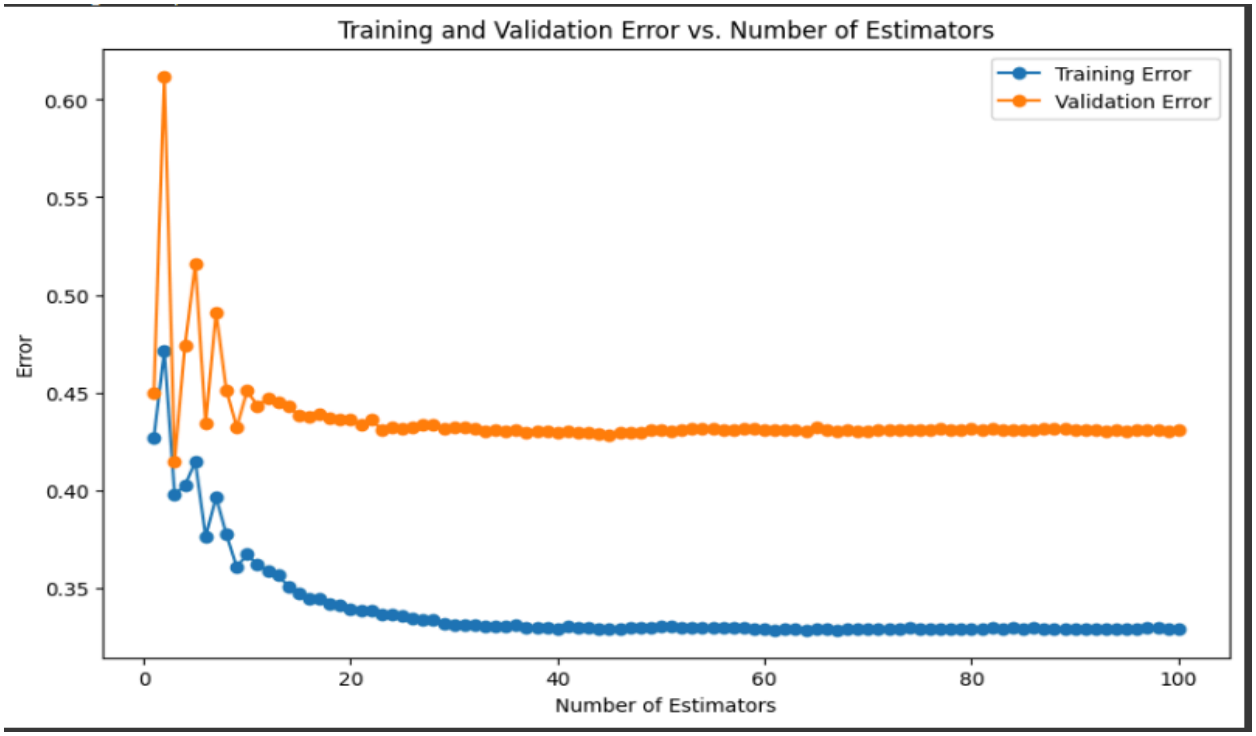
- Features with consistently low importance could be removed or combined with other features to reduce computational cost and potential noise.

- 

Explainability:

- Understanding feature importance helps interpret the model's behavior and ensures its predictions are based on meaningful attributes, which is crucial for a healthcare-related dataset like depression data.

ACCURACY- 56.91%



Key Observations:

1. **Training Error:**
   - The blue curve represents the training error, which consistently decreases as the number of estimators increases.
   - This indicates that as more weak learners (decision stumps) are added, the model becomes increasingly adept at capturing patterns in the training dataset.
   - However, after around **40 estimators**, the training error plateaus, meaning additional estimators no longer significantly improve performance on the training data.
2. **Validation Error:**
   - The orange curve represents the validation error, which initially fluctuates but stabilizes after approximately **20 estimators**.
   - The consistent gap between training and validation error suggests the model generalizes reasonably well, although some overfitting might occur if the gap widens with more estimators.
3. **Convergence:**
   - Both training and validation errors stabilize around the **30-40 estimator** mark. This implies that increasing the number of estimators beyond this point may not yield significant improvements and could lead to computational inefficiency.

TECHNIQUE- Model Ensemble Approach: Voting Classifier

- The code implements **ensemble learning** by combining the predictions of three individual models:
    - **Logistic Regression**
    - **Decision Tree Classifier**
    - **Gaussian Naive Bayes**

    - **Voting Classifier:** Combines predictions from these base models using **soft voting**, which averages the predicted probabilities to make a final decision. This often improves overall model performance by leveraging the strengths of each base model.

2. Preprocessing Steps

- **Balanced Dataset:** The model is trained on a **balanced dataset** (using SMOTE or other resampling techniques) to address class imbalance, ensuring fair representation of both classes.
- **Scaling:** The input features are normalized using a **StandardScaler**, which is crucial for models like Logistic Regression and Naive Bayes to perform effectively.

3. Voting Strategy

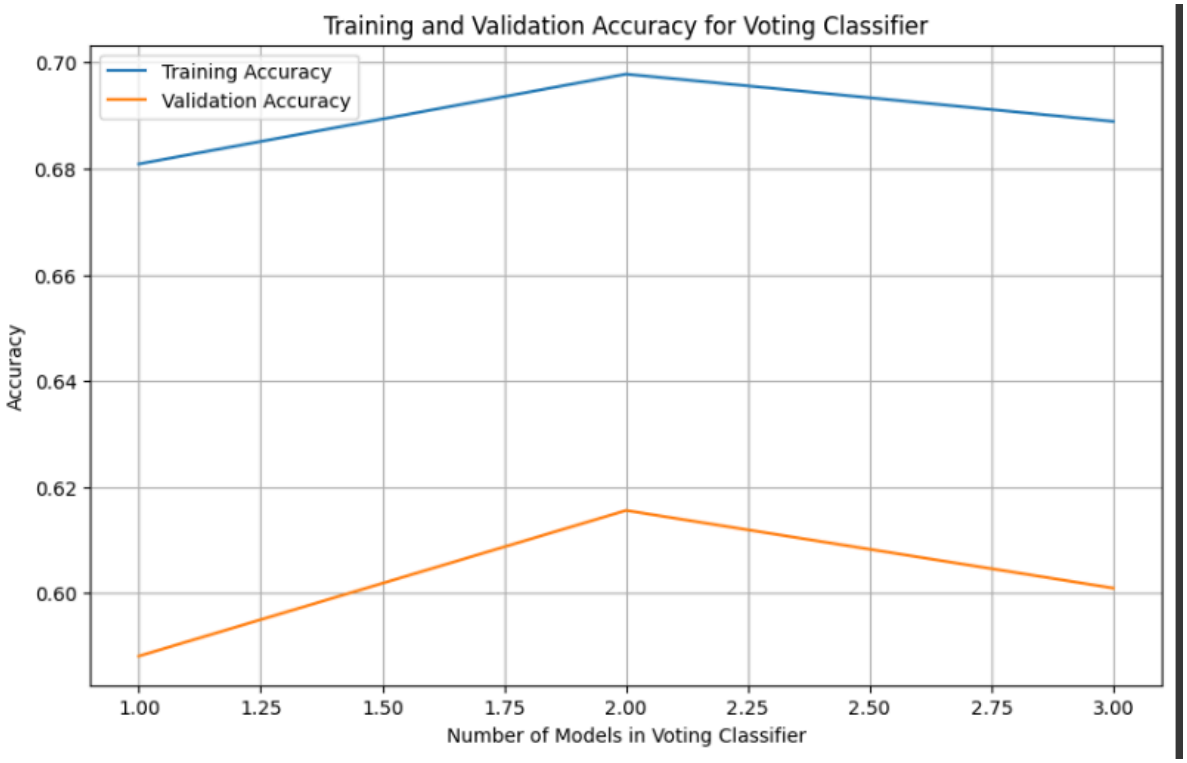
- **Soft Voting:** Averages the class probabilities predicted by individual models, which typically provides better performance than hard voting (based on majority voting).
- This approach ensures that models with high confidence in their predictions have a stronger influence on the final decision.

4. Hyperparameters

- **Logistic Regression:** `class_weight='balanced'` accounts for class imbalance, and `max_iter=1000` ensures sufficient iterations for convergence.
- **Decision Tree:** `max_depth=5` limits the tree's depth, controlling overfitting and complexity.
- **Gaussian Naive Bayes:** Assumes a Gaussian distribution for continuous features.

VOTING CLASSIFIER ACCURACY: 60.10%

```
Confusion Matrix:
 [[32996  8638]
 [16128  4303]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.79      0.73     41634
        True       0.33      0.21      0.26     20431

    accuracy                           0.60     62065
   macro avg       0.50      0.50      0.49     62065
weighted avg       0.56      0.60      0.57     62065
```



Key Observations:

1. **Training Accuracy:**
    - The training accuracy increases as more models are added to the Voting Classifier, peaking when all three models are included.
    - After reaching the maximum accuracy with two models, there is a slight drop when the third model is included. This may indicate overfitting or conflicting model predictions in the ensemble.
2. **Validation Accuracy:**

- The validation accuracy follows a similar trend, increasing with the addition of the first two models but then slightly declining when the third model is included.
- The peak validation accuracy is observed when **two models** are included in the ensemble, likely indicating the optimal combination of diverse models that generalize well.

XGBOOST Ensemble

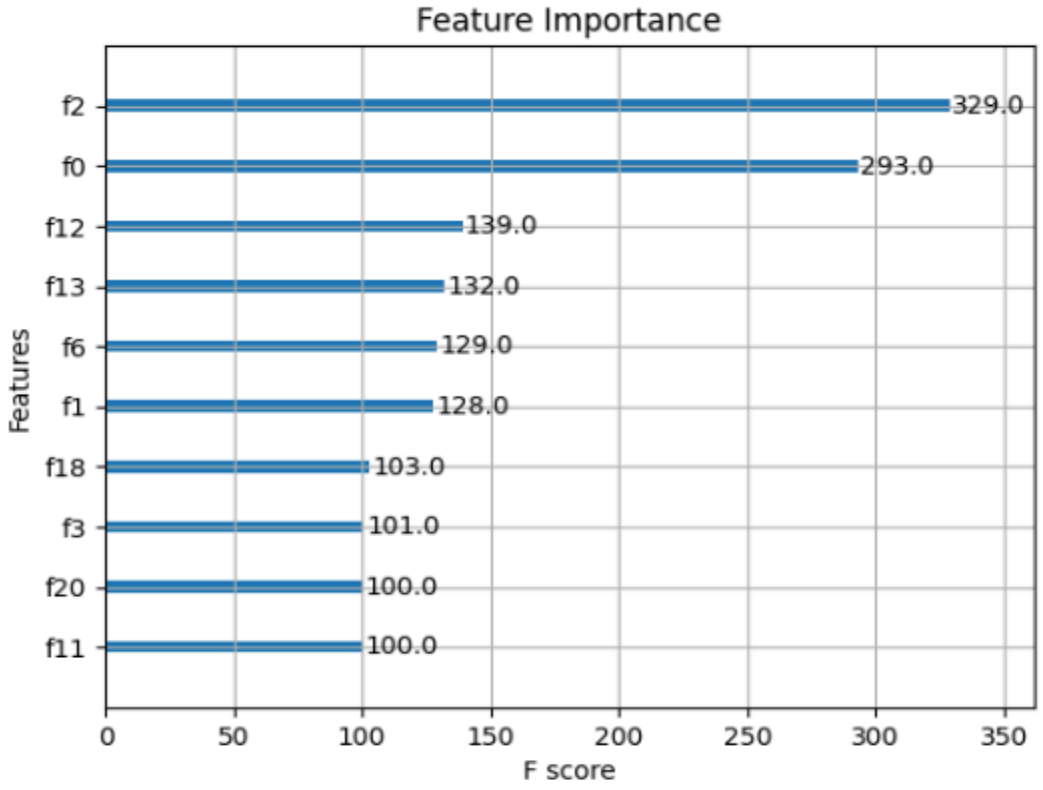
ACCURACY-66.78%

```
Confusion Matrix:
 [[41128   506]
 [20112   319]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.99      0.80     41634
        True       0.39      0.02      0.03     20431

    accuracy                           0.67     62065
   macro avg       0.53      0.50      0.41     62065
weighted avg       0.58      0.67      0.55     62065
```

XGBoost Classifier Initialization:

- **`n_estimators=200`:** Specifies 200 boosting rounds (trees) to build the ensemble.
- **`max_depth=4`:** Limits the depth of each decision tree to prevent overfitting while capturing complex patterns.
- **`learning_rate=0.1`:** Determines the step size for adjusting weights to reduce errors.
- **`colsample_bytree=0.8` and `subsample=0.8`:** Controls the fraction of columns and rows sampled for each tree, adding randomness to enhance generalization.
- **`eval_metric="logloss"`:** Optimizes the logarithmic loss function to handle imbalanced datasets effectively.
- **`random_state=42`:** Ensures reproducibility.

Analysis of the Feature Importance Plot



Top Contributing Features:

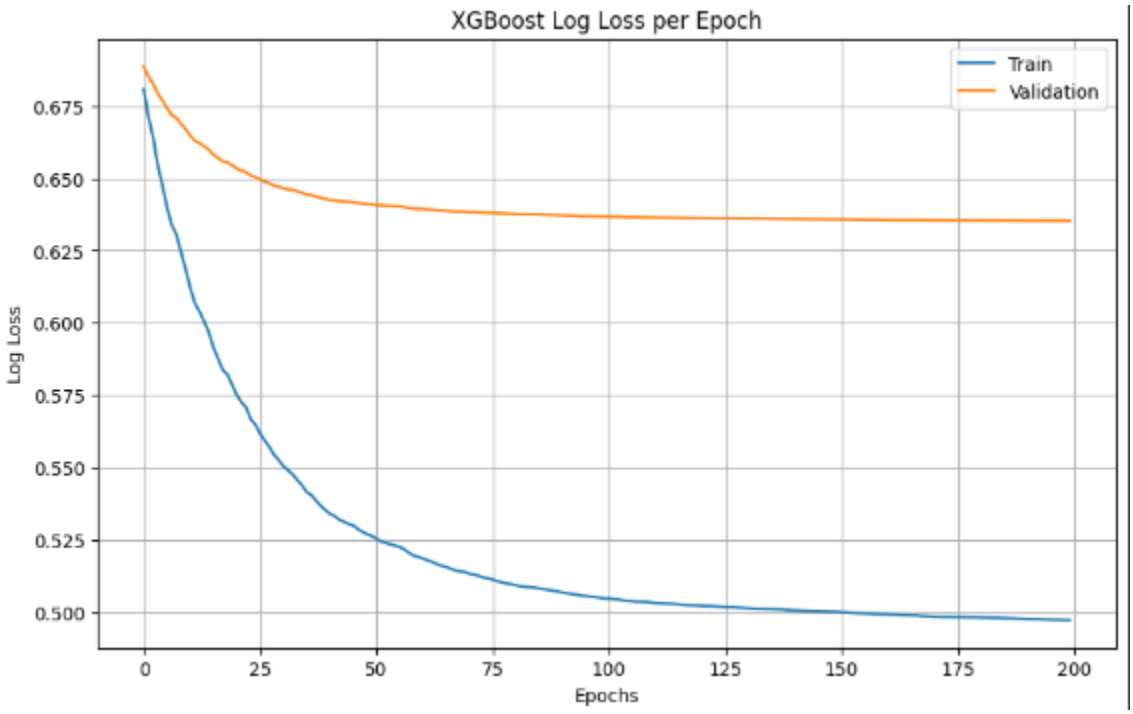
- **`f2` and `f0` are the most influential features**, with F-scores of 329 and 293, respectively. These features are used most frequently for decision splits in the trees.
- This indicates that the dataset's predictive power is highly concentrated in these features.

Significant but Secondary Features:

- Features like `f12`, `f13`, `f6`, and `f1` have moderate importance scores (ranging from ~128 to ~139).
- These features contribute meaningfully to the model but are not as impactful as `f2` and `f0`.

Low-Contributing Features:

- Features like `f18`, `f3`, `f20`, and `f11` show relatively low importance (scores ~100). While they still influence the model, their contribution is limited compared to the top features.



Analysis of the XGBoost Log Loss per Epoch Curve

1. **Observations:**
   - The **training log loss (blue curve)** decreases sharply during the initial epochs, indicating that the model is quickly learning patterns from the training data.
   - The **validation log loss (orange curve)** decreases initially but then plateaus after about 50 epochs, showing that the model has reached its optimal performance on the validation set.
2. **Model Performance:**
   - The steady decline in training log loss reflects the increasing accuracy of predictions on the training set.
   - The plateauing of the validation log loss suggests that further training does not significantly improve generalization and might lead to overfitting if continued.

Linear Discriminant Analysis (LDA)

ACCURACY: 59.95%

```
Confusion Matrix (LDA):
 [[32952  8682]
 [16173  4258]]
Classification Report (LDA):
              precision    recall  f1-score   support

       False       0.67      0.79      0.73     41634
        True       0.33      0.21      0.26     20431

    accuracy                           0.60     62065
   macro avg       0.50      0.50      0.49     62065
weighted avg       0.56      0.60      0.57     62065
```

Purpose: LDA is a classification technique that works by finding a linear combination of features that best separates the classes. It assumes that the data from each class has a Gaussian distribution with identical covariance matrices.

Key Steps in Code:

- The `LinearDiscriminantAnalysis()` model is initialized and trained on the **balanced and scaled training data** (`X_train_balanced_scaled` and `y_train_balanced`).
- The model predicts class labels for the validation set (`X_val_scaled`).

Quadratic Discriminant Analysis (QDA):

ACCURACY: 61.76%

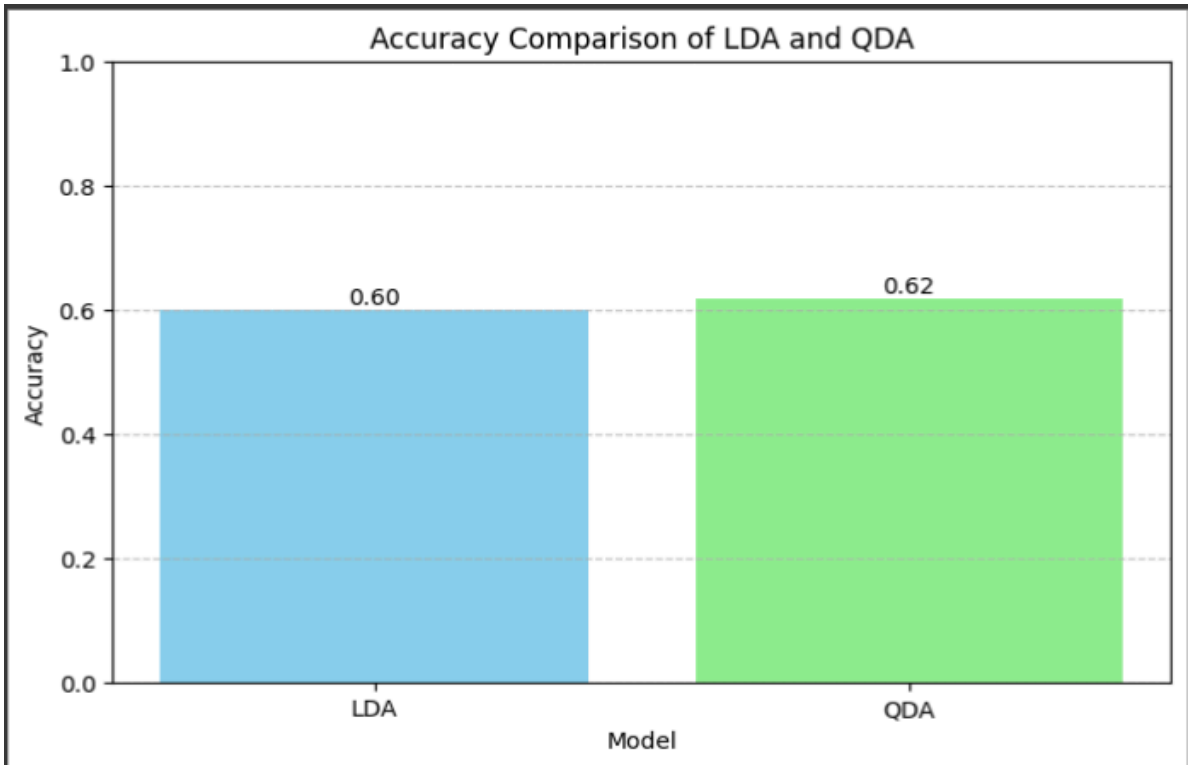
```
Confusion Matrix (QDA):
 [[34127  7507]
 [16229  4202]]
Classification Report (QDA):
              precision    recall  f1-score   support

       False       0.68      0.82      0.74     41634
        True       0.36      0.21      0.26     20431

    accuracy                           0.62     62065
   macro avg       0.52      0.51      0.50     62065
weighted avg       0.57      0.62      0.58     62065
```

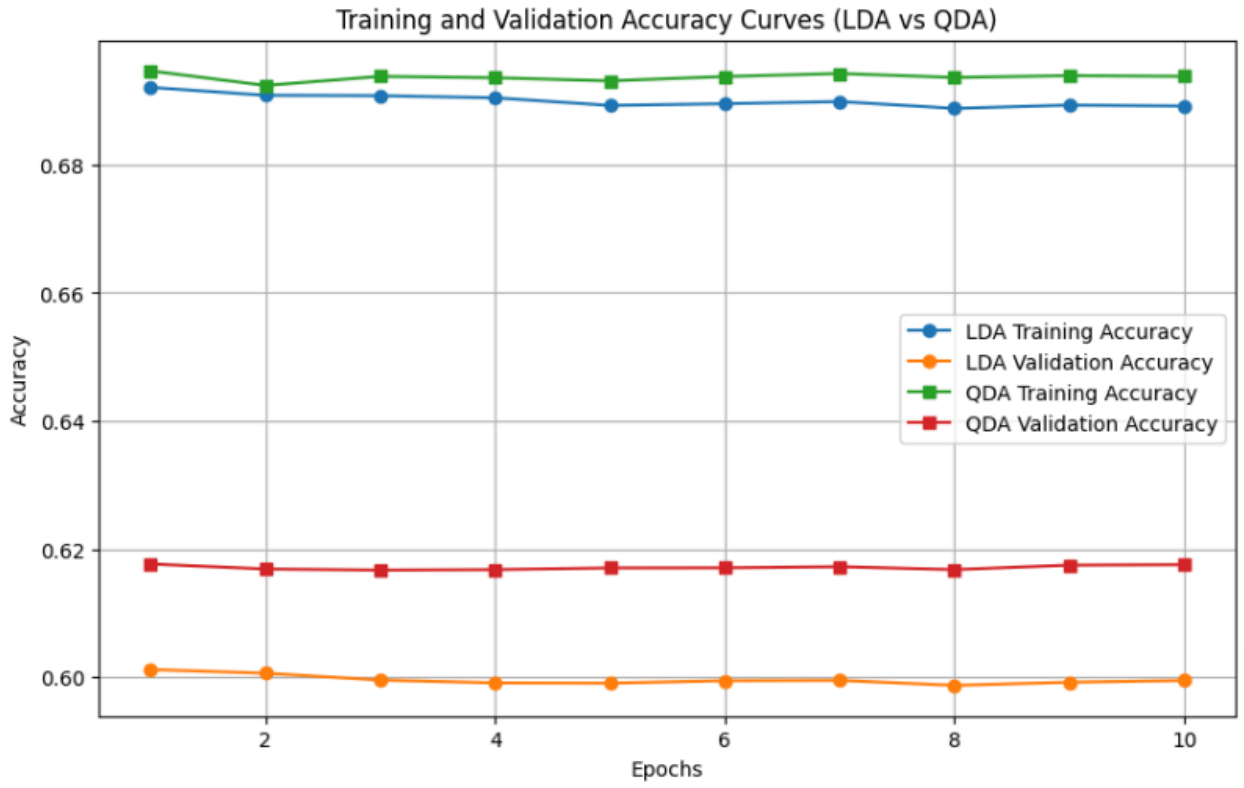
- **Purpose:** QDA is an extension of LDA that allows each class to have its own covariance matrix, making it suitable for problems with non-linear boundaries.
- **Key Steps in Code:**
    - The `QuadraticDiscriminantAnalysis()` model is initialized and trained on the same balanced and scaled training data.
    - Predictions and evaluation metrics are calculated similarly to LDA, allowing for a comparison of their performances.
- **Key Feature:** QDA introduces flexibility to model non-linear class separations, making it useful for more complex datasets.

LDA: Works well when the dataset has linear separability and equal class covariances.

QDA: Handles non-linear class boundaries but requires more data due to higher complexity.

Accuracy Comparison of LDA vs QDA





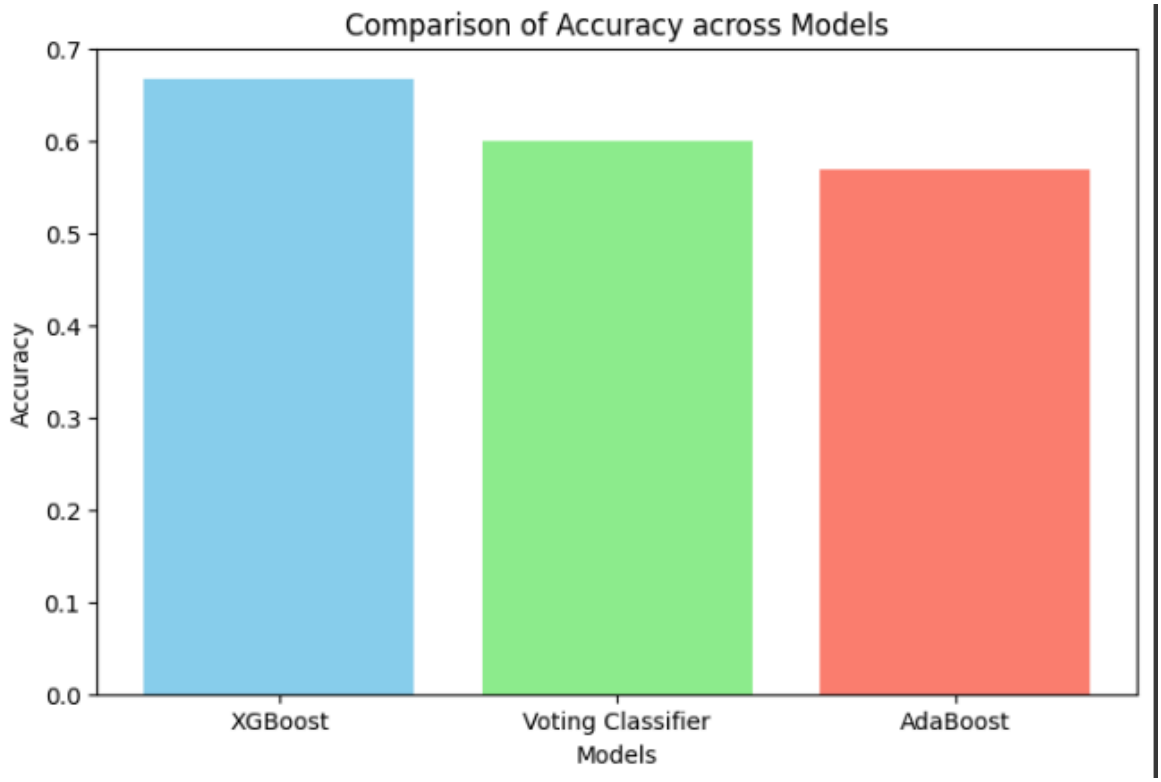
LDA Performance:

- **Training Accuracy**: The curve for LDA training accuracy is consistently high, hovering around 68.5%. This indicates that LDA is performing well on the training data.
- **Validation Accuracy**: The validation accuracy is slightly lower, staying around 60%. The gap between training and validation accuracy suggests that LDA might slightly overfit the training data but still generalizes decently.

QDA Performance:

- **Training Accuracy**: QDA training accuracy is consistent and lower than LDA, at around 62%. This reflects that QDA is less capable of fitting the training data, likely due to its increased complexity and the requirement for larger datasets.

- **Validation Accuracy**: The QDA validation accuracy remains steady around 60%, similar to its training accuracy. This suggests that QDA is less prone to overfitting compared to LDA but may struggle to capture the data's structure effectively.



Inter-Techniques Analysis:

1. **Voting Classifier:**
    - Accuracy: **60.1%**
    - Strengths: Combines predictions from multiple models (Logistic Regression, Decision Tree, and Naive Bayes) using soft voting, leveraging their strengths.
2. **Linear Discriminant Analysis (LDA):**
    - Accuracy: **59.95%**
    - Strengths: Performs well with linearly separable data and balances precision and recall fairly well.
3. **Quadratic Discriminant Analysis (QDA):**
    - Accuracy: **61.8%**
    - Strengths: Outperforms LDA and the Voting Classifier by capturing some non-linear relationships. Precision is higher compared to LDA, though recall remains low.
4. **XGBoost:**
    - Accuracy: **66.8%**
    - Strengths: Gradient boosting with optimized decision trees leads to better performance than other techniques. Particularly excels in leveraging relationships across features for improved accuracy.

5. **AdaBoost:**
    - Accuracy: **56.9%**
    - Strengths: Boosts weak learners iteratively, showing decent recall compared to other methods like XGBoost.

Best Performing Technique: XGBoost (Accuracy: 66.8%)

- **Reasoning:**
    1. **Gradient Boosting Strengths:** XGBoost effectively handles large, high-dimensional datasets and learns complex relationships through sequential boosting.
    2. **Hyperparameter Optimization:** Tuning parameters such as learning rate, max depth, and subsampling allows for a better fit to the data.
    3. **Feature Importance:** XGBoost effectively identifies and utilizes the most significant features for classification.
    4. **Adaptability:** Handles missing values and categorical encoding better than techniques like LDA or QDA.

MULTI-LAYER PERCEPTRON(MLP)

Hidden Layers: Two layers with 100 and 50 neurons, respectively, allowing the model to capture complex patterns in the data.
Activation Functions: Defines how neurons fire and introduce non-linearity in the model. Options include:

- `identity`: Linear activation (no transformation).
- `logistic`: Sigmoid function, suitable for probabilistic outputs.
- `tanh`: Hyperbolic tangent, outputs between -1 and 1.
- `relu`: Rectified Linear Unit, introduces sparsity and faster training.

Optimizer: `adam` (adaptive moment estimation), which adjusts learning rates dynamically for faster convergence.

MLP with Different Activation functions:

```
Training MLP with activation function: identity
MLP Classifier Metrics: {'Accuracy': 0.58709417546121,
Confusion Matrix:
 [[31578 10056]
 [15571  4860]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.76      0.71     41634
        True       0.33      0.24      0.27     20431

    accuracy                           0.59     62065
   macro avg       0.50      0.50      0.49     62065
weighted avg       0.56      0.59      0.57     62065
```

```
Training MLP with activation function: logistic
MLP Classifier Metrics: {'Accuracy': 0.6683154757109482,
Confusion Matrix:
 [[41325   309]
 [20277   154]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.99      0.80     41634
        True       0.33      0.01      0.01     20431

    accuracy                           0.67     62065
   macro avg       0.50      0.50      0.41     62065
weighted avg       0.56      0.67      0.54     62065
```

```
Training MLP with activation function: tanh
MLP Classifier Metrics: {'Accuracy': 0.6671070651736083
Confusion Matrix:
 [[41037   597]
 [20064   367]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.99      0.80     41634
        True       0.38      0.02      0.03     20431

    accuracy                           0.67     62065
   macro avg       0.53      0.50      0.42     62065
weighted avg       0.58      0.67      0.55     62065
```

```
Training MLP with activation function: relu
MLP Classifier Metrics: {'Accuracy': 0.6670587287521147
Confusion Matrix:
 [[41067   567]
 [20097   334]]
Classification Report:
              precision    recall  f1-score   support

       False       0.67      0.99      0.80     41634
        True       0.37      0.02      0.03     20431

    accuracy                           0.67     62065
   macro avg       0.52      0.50      0.42     62065
weighted avg       0.57      0.67      0.55     62065
```

Reasons Why MLP with Non-linear Activation Functions and XGBoost are the Best Performing Models:

1. **MLP with Non-linear Activation Functions (67% Accuracy):**
    - **Non-linearity in Activation Functions:**
        - Non-linear activation functions like `relu`, `tanh`, and `logistic` enable the model to learn complex relationships in the data that linear models cannot capture.
        - `ReLU` specifically helps in faster convergence by avoiding the vanishing gradient problem and creating sparsity in activations.
    - **Multi-layer Architecture:**

        - The two hidden layers (100 and 50 neurons) allow MLP to learn hierarchical representations of features, capturing both low-level and high-level patterns in the data.
    - **Adaptive Optimization (Adam):**
        - The `adam` optimizer dynamically adjusts the learning rate, leading to faster convergence and robustness across different activation functions and datasets.
    - **Versatility with Features:**
        - MLP is capable of learning non-linear relationships in feature interactions, which is particularly useful in complex datasets like this one, where dependencies between features may not be linear.

XGBoost (67% Accuracy):

- **Boosting Framework:**
    - XGBoost uses **gradient boosting**, an iterative process where each tree corrects the errors of the previous ones, leading to a highly accurate ensemble of decision trees.
- **Feature Importance Utilization:**
    - XGBoost assigns feature importance scores during training, effectively focusing on the most impactful features, reducing noise and enhancing predictive power.
- **Regularization and Pruning:**
    - Regularization parameters prevent overfitting, while the max-depth of 4 and learning rate of 0.1 ensure that the trees are shallow enough to generalize well while learning fine-grained patterns.
- **Column Subsampling and Row Subsampling:**
    - By using `colsample_bytree=0.8` and `subsample=0.8`, XGBoost reduces variance and avoids overfitting by training trees on random subsets of features and data points.

Why These Models Performed Well:

1. **Handling Non-linear Relationships:**
    - Both models excel at learning non-linear relationships and feature interactions, which are common in real-world datasets like this one.

2. **Feature Importance:**
    - XGBoost's feature importance mechanism and MLP's ability to learn hierarchical patterns allowed them to leverage the most impactful features effectively.
3. **Regularization:**
    - Both models have in-built mechanisms to prevent overfitting (e.g., early stopping in MLP, regularization in XGBoost), ensuring robust performance on unseen data.
4. **Balancing Techniques:**
    - The use of SMOTE ensured that both models focused equally on minority and majority classes, enhancing recall and precision.